A PROCESS FOR MANAGING SYSTEM STACKS IN MICROCONTROLLERS, CORRESPONDING DEVICE AND COMPUTER PROGRAM PRODUCT

BACKGROUND OF THE INVENTION

Field of the invention

The present invention relates to the management of system stacks in microcontrollers and has been developed with particular attention paid to its possible application to the management of a system stack in a microcontroller, for example of the register-file-based type. Reference to the above possible application must not, however, be understood as in any way limiting the sphere of protection of the invention, which is of an altogether general nature.

Description of the Related Art

A typical operation that is very frequently carried out in a microcontroller is the storage/restoring of the state of the system in particular locations of the volatile memory (generally RAM). The above operation derives directly from the need to perform jumps to special routines at arbitrary instants in time, within the program flow, following upon occurrence of external or internal interrupts.

The state of the system consists of a certain number of registers, which is different for each microcontroller. Typically, the state can be reconstructed using the registers Program Counter (PC) and Condition Code Register (CCR), in addition to a variable number of other registers.

The operation of storage/restoring of the state may thus occupy the microcontroller for a time interval dedicated to the context-switching function. The said interval is usually quite long, owing to the sequentiality with which writing/restoring of the state in the system stack takes place. This determines a degradation in the overall performance, also in control applications that are not real-time ones.

Furthermore, very frequently the registers do not have dimensions equal to multiples of the length of the word of the volatile memory in which they are to be stored. A typical example of such a situation is the storage of the CCR, which normally consists of just a few bits.

The situation in a RAM following upon one or more executions of different interrupts may therefore be the one represented – purely by way of example – in Figure 1 of the annexed drawings, where the reference U designates unused memory locations.

The situation described above amounts to a waste of memory due to the small dimension of the CCR, or of any other state register that has a size that is small as compared to the length of the RAM word.

If a traditional solution is resorted to, such as the one illustrated by way of example in Figure 2, management of the system stack within the RAM, designated by ST, takes place in a sequential way, without optimization of the area used by the system stack in the memory.

The traditional architecture represented in Figure 2 envisages that a control unit UC will generate two signals use_ss and up_down addressed to the unit SM functioning as stack manager. The unit SM generates the pointer indicating the system stack, which is commonly referred to as System Stack Pointer or ssp, which is to be forwarded to a first module 10.

The module 10 functions as a multiplexer for selecting between the signal ssp – during context switching – and the signal ram_pointer, indicating the RAM address – during normal operation –, so as to generate a signal address_ram used for managing the portion of RAM dedicated to the addresses.

Another module 12, having the function of a multiplexer, receives at input, in addition to the signal data_bus coming from the data bus of the microcontroller, also two signals PC and CCR indicating the Program Counter and the Condition Code Register, respectively, the purpose being to generate at output the signal data_ram used for managing the portion of RAM dedicated to the data.

The references 10a and 12a designate the lines on which the signals for selecting the multiplexers regulating operation of the modules 10 and 12 are present, whilst the lines designated by wen and csn correspond to the lines on which the corresponding signals indicating, respectively, write enable and read enable of the RAM where the system stack resides are forwarded to the stack ST.

Present on the output line of the stack, designated by 14, is the corresponding output signal, out_stack.

The architecture represented in Figure 2 corresponds to solutions of a known type, this fact rendering any further detailed description thereof superfluous.

The above solution is used, for example, in the microcontroller sold under the trade name ST 5 by STMicroelectronics, the assignee of the present application. This is a microcontroller of the type referred to as register-file-based.

Execution of the context-switching operation mentioned previously typically requires a time interval of approximately two clock cycles for a microcontroller of the ST 5 type and approximately ten clock cycles for a microcontroller of the accumulator-based type.

SUMMARY OF THE PRESENT INVENTION

An embodiment of the present invention overcomes the drawbacks presented by the solution according to the prior art referred to previously, so reducing both the waste of memory and the execution times for context switching.

The embodiment employs a procedure for stack management having the characteristics referred to specifically in the claims that follow.

The invention also relates to the corresponding device, as well as the corresponding computer product, which can be directly loaded into the internal memory of a computer and comprises portions of software code that can implement the procedure according to the present invention when the product is run on a computer.

In the currently preferred embodiment, the solution according to the present invention envisages the use of two enabling signals for two different stack managers. Recourse to this solution means that, in the case where the microcontroller has to execute, for example, a call instruction, it is necessary to store/restore a smaller amount of information. Added to this is the possibility of not having to gain access to one of the two stacks. For an instruction such as a call instruction it is not necessary, in fact, to store the CCR.

The fact of having two stacks to which access may be gained in a concurrent way improves performance in terms of the time required for context switching, in effect halving it. In addition, the possibility of using different types of memory to contain the system stack also enables improvement of use of the resources: stacks made with memory elements the row dimensions of which may be different from one another reduce the waste of locations, that would otherwise remain unused.

The solution according to the present invention is of general application, since it can be applied, for example, both to accumulator-based microcontrollers and to register-file-based microcontrollers. The solution according to the present invention is, moreover, independent of the number of registers stored during context switching and reduces the corresponding time, so improving overall performance of the microcontroller. In addition, the solution according to the present invention optimizes the use of the memory for the system stack.

BRIEF DESCRIPTION OF THE ANNEXED DRAWINGS

The invention will now be described, purely by way of non-limiting example, with reference to the annexed drawings, in which:

Figures 1 and 2, which regard the known art, have already been described above; and

Figure 3 illustrates, with a formalism that aims at enabling direct comparison with the known solution illustrated in Figure 2, an architecture for management of a system stack operating according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The example embodiment of the invention represented in Figure 3 is essentially based upon the solution of splitting into two the architecture represented in Figure 2.

Albeit maintaining the presence of a control unit UC for generating the signal up_down, which discriminates between a "push" operation and a "pop" operation (where the terms appearing in quotation marks are currently used in the relevant technical field), the presence is envisaged of two distinct task managers, designated by SM1 and SM2. The control unit UC sends to the managers SM1 and SM2 two signals use_ss1 and use_ss2, respectively, which issue to the manager SM1 and to the manager SM2 the order to execute a push operation or a pop operation.

Basically, the upper branch of the architecture represented in Figure 3, which comprises the stack manager SM1, can be approximately assimilated – as regards its general structure – to the known structure represented in Figure 2. For this reason, in the top part of Figure 3, to designate signals or parts that are identical or equivalent to the ones already described previously with reference to the prior art, the same references already appearing in Figure 2 are used. It is, therefore, superfluous to repeat here the corresponding description already provided previously.

It will, however, be appreciated that a different element, in the diagram of Figure 3, is that only the signal PC coming from the Program Counter is sent to the module 12, together with the signal data_bus, and not the signal CCR representing the Condition Code Register.

In addition, in the diagram of Figure 3, the signal ssp which represents the pointer indicating the system stack is, in effect, split into two components sspl and ssp2 generated by the stack manager SM1 and the stack manager SM2, respectively.

Operation of the scheme represented in Figure 3 envisages that the two stack managers SM1 and SM2, driven by the control unit UC, will work in parallel so that:

one manager, SM1, will manage a first stack ST1 within a memory, such as a RAM, used in a conventional way, and

the other manager, SM2, will manage a second stack ST2 consisting of a bank of memory elements (basically, flip-flops) purposely created and equal in number to the number of bits of the signal CCR representing the Condition Code Register times the number of the interrupts of the microcontroller.

The stack managers SM1 and SM2 operate by generating two stack pointer signals (namely, sspl and ssp2), the value of which is always equal to the first memory location available within the respective stack ST1 or ST2.

In particular, in the ensuing description, it will be assumed that, corresponding to each push operation on the stacks, there corresponds a decrement of the stack pointer associated therewith. In a complementary way, corresponding to each pop operation is an increment of the associated stack pointer.

The control unit UC thus generates three signals for driving the two stack manager modules SM1 and SM2:

a first signal up_down, the purpose of which is to discriminate between a push operation and a pop operation;

a signal use_ss1, the purpose of which is to enable explicitly the push operation or the pop operation on the first stack ST1, and a signal use_ss2 the purpose of which is to explicitly enable the push operation or the pop operation on the second stack ST2.

When an interrupt occurs, the control unit UC sets the signal up_down so as to obtain a push operation. Simultaneously, it drives the write signals of the two stacks and sets the signals use_ss1 and use_ss2 so as to indicate to the stack managers to decrement their own stack pointers in order to point to the next available location.

In a complementary way, for restoring the state at the end of the interrupt routine, the control unit UC sets the signal up_down so as to obtain a pop operation. Simultaneously, it drives the signals use_ss1 and use_so as to indicate to the managers to increment their own stack pointers in order to point to the next location containing the stored state which is to be restored. Finally, it sets the signals for the reading of the two stacks.

As regards the connections of the two stacks in question, it will be appreciated that the connections of the stack ST1 are practically identical to that of the single stack ST represented in Figure 2, which regards the prior art.

The stack ST2, which is responsible for storage of the signal CCR, receives instead at input, in addition to the signal ssp2 already described previously, a signal data_stack_ccr, which indicates the value of the Condition Code Register, as well as the two signals, the enable signal and the latch signals, the function of which is to enable reading and writing, respectively, of the stack ST2.

The references 14a and 14b designate the corresponding output lines of the stacks ST1 and ST2, in which the corresponding read output signals are present.

If the traditional architecture represented in Figure 2 is adopted, both the stage of writing in the system stack and the stage of reading from the system stack entail two clock cycles, one dedicated to the Program Counter and the other to the signal CCR. If, instead, recourse is had to the parallel architecture represented in Figure 3, both the stage of writing in the stack and the stage of reading from the stack can be carried out in a parallel way for the Program Counter and for the CCR, just occupying a single clock cycle.

All of the above U.S. patents, U.S. patent application publications, U.S. patent applications, foreign patents, foreign patent applications and non-patent publications referred to in this specification and/or listed in the Application Data Sheetare incorporated herein by reference, in their entirety.

Of course, without prejudice to the principle of the invention, the details of implementation and the embodiments may be amply varied with respect to what is described and illustrated herein, without thereby departing from the scope of the present invention, as defined in the annexed claims.